3.1 第 1 关：基本测试

根据 S-DES 算法编写和调试程序，提供 GUI 解密支持用户交互。输入可以是 8bit 的数据和 10bit 的密钥，输出是 8bit 的密文。

3.2 第 2 关：交叉测试

考虑到是**算法标准**，所有人在编写程序的时候需要使用相同算法流程和转换单元(P-Box、S-Box 等)，以保证算法和程序在异构的系统或平台上都可以正常运行。设有 A 和 B 两组位同学(选择相同的密钥 K)；则 A、B 组同学编写的程序对明文 P 进行加密得到相同的密文 C；或者 B 组同学接收到 A 组程序加密的密文 C，使用 B 组程序进行解密可得到与 A 相同的 P。

采用相同主密钥 1010000010，对明文 01000011 以及密文 01000100 进行测试。

加密：

解密：

3.3 第 3 关：扩展功能

考虑到向实用性扩展，加密算法的数据输入可以是 ASII 编码字符串(分组为 1 Byte)，对应地输出也可以是 ACII 字符串(很可能是乱码)。

3.4 第 4 关：暴力破解

假设你找到了使用相同密钥的明、密文对(一个或多个)，请尝试使用暴力破解的方法找到正确的密钥 Key。在编写程序时，你也可以考虑使用多线程的方式提升破解的效率。请设定时间戳，用视频或动图展示你在多长时间内完成了暴力破解。

```
public void actionPerformed(ActionEvent arg0) {
    //蛮力破解求主密钥
    ArrayList<String> arrayList=new ArrayList<>();
    long t1 = new Date().getTime();
    for(int i=0;i<1024;i++) {
        arrayList.add(getKey(i));
    }
    Iterator<String> iterator = arrayList.iterator();
```

开始穷举前获取一个时间点

```
    }
      mainkey.setText(sr);
    long t2 = new Date().getTime();
    long ti = t2 - t1;
    fTime.setText(ti+" 毫秒");
```

找到所有可能的主密钥后再获取一个时间点，前后相减得到消耗的时间、

3.5 第5关：封闭测试

根据第4关的结果，进一步分析，对于你随机选择的一个明密文对，是不是有不止一个密钥Key？进一步扩展，对应明文空间任意给定的明文分组 P_n ，是否会出现选择不同的密钥 $K_i \neq K_j$ 加密得到相同密文 C_n 的情况？

(1) 对于随机选择的一个明密文对，实际上存在不止一个密钥 Key。假设明文为*[1,0,0,1,1,0,1,0]，密文为[1,1,1,0,1,1,1,1]，通过暴力破解找到 2 个密钥：['1010000010', '1110000010']*，用时 2 毫秒。
(2) 进一步扩展，对于明文空间内任意给定的明文分组 Pn，会出现选择不同的密钥 Ki 和 Kj（Ki≠ Kj）加密得到相同密文 Cn 的情况。例设明文分组为*[1,0,1,0,1,0,1,0]，密文分组为[1,0,1, 0, 1, 0,1,0]， 通过暴力破解找到 8 个密钥：[0, 0, 0, 1, 0, 1, 0, 0, 0, 1] [0, 0, 1, 0, 1, 0, 1, 0, 0, 0] [0, 1, 0, 1, 0, 1, 0, 0, 0, 1] [0, 1, 1, 0, 1, 0, 1, 0, 0, 0] [1, 0, 0, 1, 1, 1, 0, 1, 1, 0] [1, 0, 1, 0, 0, 0, 1, 1, 1, 1] [1, 1, 0, 1, 1, 1, 0, 1, 1, 0] [1, 1, 1, 0, 0, 0, 1, 1, 1, 1]，用时 3 毫秒。